

# CS 312: Algorithm Analysis



## Lecture #13: Exponentiation and Cryptography

### Objectives

#### Lecture #12: Matrix Multiplication, Phylogenetics

- Apply divide and conquer to matrix multiplication
- Derive a bound using equation 7.1
- Apply divide and conquer to phylogenetics
- Derive a bound

#### Lecture #13: Exponentiation and Cryptography

- See another problem with a D/C solution.
- See a problem without a D/C solution.
- This time they are problems you care about.
- Computer Security

### Sequential Exponentiation

```

function expoSeq (a, n)
    r ← a
    for i ← 1 to n-1 do
        r ← a x r
    return r
    
```

*a<sup>n</sup>*

### D/C Exponentiation

```

function expoDC (a, n)
    if n = 1 then return a
    if n is even then
        return [expoDC (a, n/2)]2
    return a x expoDC (a, n-1)
    
```

$$a^n = \begin{cases} a & n = 1 \\ (a^{n/2})^2 & n \text{ is even} \\ a \times a^{n-1} & n \text{ is odd} \end{cases}$$

### Cost of Exponentiation

*lg = log<sub>2</sub>*

	Integer Multiplication	
	classic	D and C
expoSeq	$\Theta(m^2 n^2)$	$\Theta(m^{lg^3} n^2)$
expoDC	$\Theta(m^2 n^2)$	$\Theta(m^{lg^3} n^{lg^3})$

Compute  $a^n$  and let  $m = \text{digits}(a)$ .

### D/C Exponentiation (Iterative)

```

function expolter (a,n)
    {computes a^n}
    i = n; r = 1; x = a
    while i > 0
        if i is odd then r = rx endif
        x = x2
        i = [i / 2]
    return r
    
```

$$5^{17} = 5 \cdot 5^{16} \text{ since } 17 = 10001_2$$

## Modular Exponentiation: $a^n \bmod z$

Some identities from number theory:

$$(x \cdot y) \bmod z = [(x \bmod z) \cdot (y \bmod z)] \bmod z$$

$$(x \bmod z)^y \bmod z = x^y \bmod z$$

```
function expomod (a,n,z)
{computes a^n mod z}
i = n; r = 1; x = a mod z
while i > 0
  if i is odd then r = r x mod z
  x = x^2 mod z
  i = i +2
return r
```

## Example.

```
3^10 mod 10
a = 3, n = 10, z = 10
i = 10, r = 1, x = 3 mod 10 = 3
x' = 3^2 mod 10 = 9
i' = 5
r' = 1 x 9 mod 10 = 9
x'' = 9^2 mod 10 = 81 mod 10 = 1
i'' = 2
x''' = 1^2 mod 10 = 1
i''' = 1
r'' = 9 x 1 mod 10 = 9
x'''' = 1
i'''' = 0
return 9
```

```
function expomod (a,n,z)
{computes a^n mod z}
i = n; r = 1; x = a mod z
while i > 0
  if i is odd then r = rx mod z
  x = x^2 mod z
  i = i +2
return r
```

## Example

$$3^{20} \bmod 10$$

Two volunteers:

Volunteer A: use *expomod* to compute it.  
Volunteer B: compute  $3^{20}$  then take mod.

## Exponentiation in Mod Arith

- The key point is that  $a^n \bmod z$  is easy
  - expomod* is in  $\Theta(\log n)$
  - In fact, it requires about  $1.5 \lg n$  multiplications "for typical  $n$ " (what does "typical" mean?)
  - exposeq* requires  $n-1$  multiplications
  - when  $a, n,$  and  $z$  are 200 digit numbers
    - Assume 1 multiplication of two 200 digit numbers takes .001 seconds
    - expomod* typically takes about 1 second
    - exposeq* would require  $10^{179}$  times the Age of the Universe!
- Only works when  $n$  is an integer.

## Cryptography

- Alice wants to send Bob a message  $m$  as a secret (encrypted) message  $c$
- Classically, Bob and Alice decide on a cipher together by sharing knowledge of a key  $k$
- They want to make the cipher hard to "break";
- i.e. difficult to get  $m$  from  $c$  without  $k$
- Is it possible to communicate privately without prior coordination?

## Public Key Cryptography

Bob makes a key public Alice



Thanks to Diffie, Hellman, and Merkle

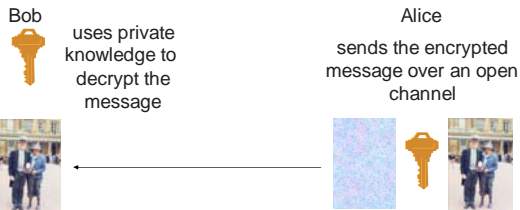
## Public Key Cryptography



## Public Key Cryptography



## Public Key Cryptography



## Public Key Cryptography

code =  $f(\text{message}, \text{public key})$   
 message =  $f^{-1}(\text{code}, \text{private key})$

- Want  $f$  to be easy to compute
- Want  $f^{-1}$  nearly impossible to compute without private key, and easy to compute with private key
- $a^n \bmod z$  is easy to compute—can we use modular arithmetic for cryptography?

## Public Key Cryptography: RSA

Suppose  $c = a^n \bmod z$ , where  $z = pq$  for two large primes  $p, q$ , and  $n$  satisfies  $1 < n < z-1$  and  $n$  shares no common factors with  $\phi = (p-1)(q-1)$ .

Note:

✓  $(x \bmod z)^y \bmod z = x^y \bmod z$

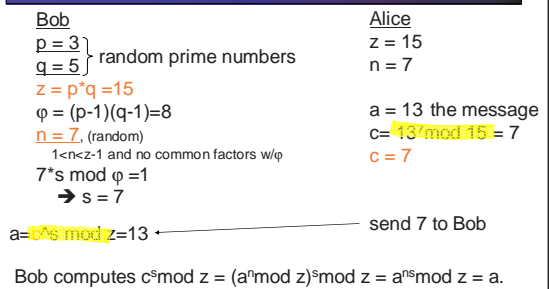
- Theorem from Number Theory:
  - $a^x \bmod z = a$  if  $0 < a < z$  and  $x \bmod \phi = 1$

Then  $c^s \bmod z = (a^n \bmod z)^s \bmod z = a^{ns} \bmod z = a$  for  $s$  is determined by  $n$  and  $\phi$  ( $p, q$ ) to satisfy  $ns \bmod \phi = 1$ .

$a$  is my message,  $c$  is the encryption,  $(n, z)$  is the public key, and  $(s, z)$  is the private key!

Thanks, Rivest, Shamir, and Adleman

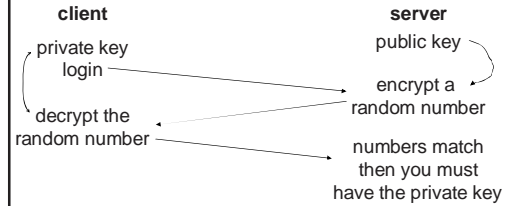
## RSA Walk-Through



## The Crux of RSA

- Bob's private knowledge is hard to compute.
- $a^x \bmod z = a$  when  $x \bmod \phi = 1$ 
  - and  $\phi = (p-1)(q-1)$  for prime numbers  $p, q$
  - and  $z = pq$ .
- Bob's private knowledge is  $p$  and  $q$ .

## RSA and SSH



## Breaking RSA

Eavesdropper intercepts  $c$ ,  $z$ , and  $n$ .

Attacks:

1. Determine  $a$  from  $c, z$  and  $n$  ( $c = a^n \bmod z$ )
  - find the  $n$ th root of  $c \bmod z$
  - hard
2. Factor  $z$  into  $p$  and  $q$  (can derive private key from public key)
  - hard

## $n$ th Root Attack

$$a = a$$

since  $a < z$ ,

$$a = a \bmod z$$

$$a = a^{n/n} \bmod z$$

$$a = (a^n \bmod z)^{1/n} \bmod z$$

definition of  $c$ .

$$a = c^{1/n} \bmod z$$

## Thoughts

- Security of this scheme remains unproven
  - Factoring large numbers into their primes may turn out to be easy or unnecessary to break the code
  - Can you break it, or prove it is hard to break?
- Are there other "one-way" functions to do the job?
- Can we approximately break the code—derive a message  $m$  within provable bounds of  $a$ ?
- Are there alternative handshaking schemes to facilitate private communication without prior coordination?
- Need creative minds on this problem (cool jobs at NSA)

## Mid-term Exam #1

### Exam:

- NOT in the Testing Center
- Take-home
- To appear on the blog Thursday morning (2/9/06)
- 3 hours, one contiguous block
- Closed book (no friends, phones, internet, computer, etc.)
- One page of notes, Notes must be stapled to exam and handed in.
- To be handed in on Monday in class